Claims:

1. A method of verifying a design, comprising the steps of:

generating a test case for execution using said design, wherein said design comprises a plurality of resources, one of said resources being required at a predetermined time to accommodate a signal during execution of said test case;

designating said one resource as an unidentified resource;

delaying binding said signal with said unidentified resource until immediately prior to said predetermined time;

thereafter binding said signal with said unidentified resource to define a bound resource; and

accessing said bound resource.

2. The method according to claim 1, further comprising the steps of:

prior to said predetermined time copying said unidentified resource to another of said resources; and

after passage of said predetermined time binding said signalwith said another resource.

3. The method according to claim 1, wherein said signal is an outcome determinative input to a Boolean function.

4. The method according to claim 3, further comprising the steps of:

designating a second one of said resources as a second unidentified resource for accommodation of a second signal that is a second input of said Boolean function, an output of said Boolean function being insensitive to said second input; and

avoiding binding said second unidentified resource with said second signal during execution of said test case.

5. A computer software product, comprising a computer-readable medium in which computer program instructions are stored, which instructions, when read by a computer, cause the computer to verify a design by the steps of:

generating a test case for execution using said design, wherein said design comprises a plurality of resources, one of said resources being required at a predetermined time to accommodate a signal during execution of said test case;

designating said one resource as an unidentified resource;

delaying binding said signal with said unidentified resource until immediately prior to said predetermined time during execution of said test case;

thereafter binding said signal with said unidentified resource to define a bound resource; and

accessing said bound resource.

6. The computer software product according to claim 5, wherein said computer is further instructed to perform the steps of:

prior to said predetermined time copying said unidentified resource to another of said resources; and

after passage of said predetermined time binding said signal with said another resource.

7. The computer software product according to claim 5, wherein said signal is an outcome determinative input to a Boolean function.

8. The computer software product according to claim 7, wherein said computer is further instructed to perform the steps of:

designating a second one of said resources as a second unidentified resource for accommodation of a second signal that is a second input of said Boolean function, an output of said Boolean function being insensitive to said second input; and

avoiding binding said second unidentified resource with said second signal during execution of said test case.

9. A verification system of verifying a design, comprising a test generator adapted to perform the steps of:

generating a test case for execution using said design, wherein said design comprises a plurality of resources, one of said resources being required at a predetermined time to accommodate a signal during execution of said test case;

designating said one resource as an unidentified resource;

delaying binding said signal with said unidentified resource until immediately prior to said predetermined time;

thereafter binding said signal with said unidentified resource to define a bound resource; and

accessing said bound resource.

10. The verification system according to claim 9, wherein said test generator is further adapted to perform the steps of:

prior to said predetermined time copying said unidentified resource to another of said resources; and

after passage of said predetermined time binding said signal with said another resource.

11. The verification system according to claim 9, wherein said signal is an outcome determinative input to a Boolean function.

12. The verification system according to claim 11, wherein said test generator is further adapted to perform the steps of:

designating a second one of said resources as a second unidentified resource for accommodation of a second signal that is a second input of said Boolean function, an output of said Boolean function being insensitive to said second input; and

avoiding binding said second unidentified resource with said second signal during execution of said test case.

13. A method of verifying a design, comprising the steps of:

generating a stream of instructions, said stream including a first instruction that references an unidentified first resource;

setting a first flag that designates said first resource as being unidentified;

copying said first resource to a second resource;

setting a second flag that designates said second resource as being unidentified;

including said second resource in a set of unidentified resources that is associated with said first resource, wherein each member of said set has a status flag designating said member as being unidentified; and

thereafter performing the steps of

identifying said second resource;

clearing said second flag; and

removing said second resource from said set.

14. The method according to claim 13, wherein said step of identifying said second resource further comprises the steps of:

identifying each said member of said set with said second resource; and

clearing said status flag of each said member.

15. The method according to claim 14, further comprising the step of accessing said second resource.

16. The method according to claim 13, further comprising the step of accessing said second resource.

17. The method according to claim 13, wherein said step of identifying said second resource comprises requesting a content of said second resource.

18. A computer software product, comprising a computer-readable medium in which computer program instructions are stored, which instructions, when read by a computer, cause the computer to perform a method of verifying a design, comprising the steps of:

generating a stream of design instructions, said stream including a first instruction that references an unidentified first resource;

setting a first flag that designates said first resource as being unidentified;

copying said first resource to a second resource;

setting a second flag that designates said second resource as being unidentified;

including said second resource in a set of unidentified resources that is associated with said first resource, wherein each member of said set has a status flag designating said member as being unidentified; and

thereafter performing the steps of

identifying said second resource;

clearing said second flag; and

removing said second resource from said set.

19. The computer software product according to claim 18, wherein said step of identifying said second resource further comprises the steps of:

identifying each said member of said set with said second resource; and

clearing said status flag of each said member.

20. The computer software product according to claim 19, further comprising the step of accessing said second resource.

21. The computer software product according to claim 18, further comprising the step of accessing said second resource.

22. The computer software product according to claim 18, wherein said step of identifying said second resource comprises requesting a content of said second resource.

23. A verification system for verifying a design, comprising a test generator adapted to perform the steps of:

generating a stream of instructions, said stream including a first instruction that references an unidentified first resource;

setting a first flag that designates said first resource as being unidentified;

copying said first resource to a second resource;

setting a second flag that designates said second resource as being unidentified;

including said second resource in a set of unidentified resources that is associated with said first resource, wherein each member of said set has a status flag designating said member as being unidentified; and

thereafter performing the steps of:

identifying said second resource;

clearing said second flag; and

removing said second resource from said set.

24. The verification system according to claim 23, wherein in said step of identifying said second resource said test generator is further adapted to identify each said member of said set with said second resource; and to clear said status flag of each said member.

25. The verification system according to claim 23, wherein immediately prior to identifying said second resource said test generator requests a content of said second resource.

26. A method of verifying a design, comprising the steps of:

generating a stream of instructions for evaluation of a Boolean function in said design;

constructing a set of inputs for said Boolean function, said set comprising members having unidentified input resources, and said inputs being outcome determinative of said Boolean function, said Boolean function further having an output resource;

selecting one of said members;

resolving an identity of said selected member;

excluding said selected member from said set of inputs;

removing all remaining members of said set of inputs that are no longer outcome determinative of said Boolean function;

iterating said steps of selecting, resolving, excluding and removing until no more than one member remains in said set of inputs; and determining said output resource as a copy of said one member.

27. The method according to claim 26, further comprising the steps of:

setting a flag that designates said output resource as being unidentified; and

including said output resource in a set of unidentified resources that is associated with said one member, respective status flags being associated with each member of said set of unidentified resources that designate an unidentified status thereof.

28. The method according to claim 27, further comprising the steps of:

associating a respective inversion flag with each of said input resources and said output resource that indicate an inversion status thereof; and

setting said inversion flag of said output resource to a negation of said inversion flag of one of said input resources that corresponds to said one member.

29. A computer software product, comprising a computer-readable medium in which computer program instructions are stored, which instructions, when read by a computer, cause the computer to perform a method of verifying a design, comprising the steps of:

generating a stream of design instructions for evaluation of a Boolean function in said design;

constructing a set of inputs for said Boolean function, said set comprising members having unidentified input resources, and said inputs being outcome determinative of said Boolean function, said Boolean function further having an output resource;

selecting one of said members;

resolving an identity of said selected member;

excluding said selected member from said set of inputs;

removing all remaining members of said set of inputs that are no longer outcome determinative of said Boolean function;

iterating said steps of selecting, resolving, excluding and removing until no more than one member remains in said set of inputs; and

determining said output resource as a copy of said one member.

30. The computer software product according to claim 29, wherein said computer is further instructed to perform the steps of:

setting a flag that designates said output resource as being unidentified; and

including said output resource in a set of unidentified resources that is associated with said one member, respective status flags being associated with each member of said set of unidentified resources that designate an unidentified status thereof.

31. The computer software product according to claim 30, wherein said computer is further instructed to perform the steps of:

associating a respective inversion flag with each of said input resources and said output resource that indicate an inversion status thereof; and setting said inversion flag of said output resource to a negation of said inversion flag of one of said input resources that corresponds to said one member.

32. A verification system of verifying a design, comprising a test generator adapted to perform the steps of:

generating a stream of instructions for evaluation of a Boolean function in said design;

constructing a set of inputs for said Boolean function, input resources associated with members of said set of inputs being unidentified, and said inputs being outcome determinative of said Boolean function, said Boolean function further having an output resource;

selecting one of said members;

resolving an identity of said selected member;

excluding said selected member from said set of inputs;

removing all remaining members of said set of inputs that are no longer outcome determinative of said Boolean function; and

iterating said steps of selecting, resolving, excluding and removing until no more than one member remains in said set of inputs.

33. The verification system according to claim 32, wherein said test generator is further adapted to perform the steps of:

determining said output resource as a copy of said one member;

setting a flag that designates said output resource as being unidentified; and including said output resource in a set of unidentified resources that is associated with said one member, a respective status flag being associated with each member of said set of unidentified resources that designates an unidentified status thereof.

34. The verification system according to claim 33, wherein said test generator is further adapted to perform the steps of:

associating a respective inversion flag with each of said input resources and said output resource that indicate an inversion status thereof; and

setting said inversion flag of said output resource to a negation of said inversion flag of one of said input resources that corresponds to said one member.

35. A verification system for a computer program-under-test, comprising:

a case generator accepting said program-under-test as a program input;

a simulator for executing said program-under-test responsive to values provided by said case generator;

a late binding component that accepts a request for user input responsively to said program-under-test, said late binding component being adapted to hold said request for user input in a memory, and for associating an unidentified resource with said request for user input;

a tracking component for determining when said unidentified resource is actually required in said simulator; and

a user interface linked to said tracking component for accepting information to satisfy said request for user input, wherein responsively to said user interface said tracking component binds said information to said unidentified resource.

36. The verification system according to claim 35, wherein said case generator is further adapted to present a context of said program-under-test on said user interface.

37. The verification system according to claim 35, wherein said case generator is further adapted to report an actual requirement for said user input in said simulator.

38. A method for verifying a computer program-under-test, comprising the steps of:

accepting a request for user input responsively to said program-under-test;

memorizing said request for user input;

associating an unidentified resource with said request for user input;

determining when said unidentified resource is actually required in an execution of said program-under-test; and

thereafter accepting information from a user to satisfy said request for user input;

binding said information to said unidentified resource to define a late-bound resource; and

using said late-bound resource in said execution of said program-under-test.

39. The method according to claim 38, further comprising the step of presenting a context of said program-under-test.

40. The method according to claim 38, further comprising the step of reporting an actual requirement for said user input in said execution of said program-under-test.

41. A computer software product, comprising a computer-readable medium in which computer program instructions are stored, which instructions, when read by a computer, cause the computer to perform a method for verifying a computer program-under-test, comprising the steps of:

defining a case generator;

submitting said program-under-test to said case generator;

accepting a request for user input responsively to said program-under-test;

memorizing said request for user input;

associating an unidentified resource with said request for user input;

determining when said unidentified resource is actually required in an execution of said program-under-test; and

thereafter accepting information from a user to satisfy said request for user input;

binding said information to said unidentified resource to define a late-bound resource; and

using said late-bound resource in said execution of said program-under-test.

42. The computer software product according to claim 41, wherein said computer is further instructed to present a context of said program-under-test.

43. The computer software product according to claim 41, wherein said computer is further instructed to report an actual requirement for said user input in said execution of said program-under-test.